

# Cooperative Game Theoretic Solution Concepts for *top-k* Problems

Swapnil Dhamal<sup>1</sup>, Akanksha Meghlan<sup>2</sup>, and Y. Narahari  
Indian Institute of Science, Bangalore, India

## Abstract

The problem of finding the  $k$  most critical nodes, referred to as the *top-k* problem, is a very important one in several contexts such as information diffusion and preference aggregation in social networks, clustering of data points, etc. It has been observed in the literature that the value allotted to a node by most of the popular cooperative game theoretic solution concepts, acts as a good measure of appropriateness of that node (or a data point) to be included in the *top-k* set, by itself. However, in general, nodes having the highest  $k$  values are not the desirable *top-k* nodes, because the appropriateness of a node to be a part of the *top-k* set depends on other nodes in the set. As this is not explicitly captured by cooperative game theoretic solution concepts, it is necessary to post-process the obtained values in order to output the suitable *top-k* nodes. In this paper, we propose several such post-processing methods and give reasoning behind each of them, and also propose a standalone algorithm that combines cooperative game theoretic solution concepts with the popular greedy hill-climbing algorithm.

**Keywords:** cooperative game theory, social networks, information diffusion, influence maximization, preference aggregation, clustering, greedy hill-climbing.

## 1 Introduction

The combinatorial problem of determining top  $k$  nodes is encountered in a number of contexts, some of which are as follows:

- *Influence maximization*: determining  $k$  seed nodes to trigger a campaign so as to maximize its spread in a social network [7]
- *Influence limitation*: determining  $k$  nodes to trigger a counter-campaign so as to limit the spread of some misinformation in a social network [2]

---

<sup>1</sup>This is a work in progress. If you have any comments, suggestions, or doubts, please send an email to [swapnil.dhamal@csa.iisc.ernet.in](mailto:swapnil.dhamal@csa.iisc.ernet.in)

<sup>2</sup>The first two authors have contributed equally.

- *Virus inoculation*: determining  $k$  nodes to vaccinate so as to restrict the spread of some epidemic in a social network [1]
- *Preference aggregation*: determining  $k$  representatives so that their aggregated preferences is close to the aggregate preference of a social network [5]
- *K-means clustering*: determining  $k$  cluster centers for initiating the K-means clustering algorithm so as to partition a dataset into  $k$  clusters in an effective way [6] (for consistency, we refer to the points in the dataset as nodes only)

In all of these problems, the input is either a network, or that can be converted to a network. For the problems of influence maximization, influence limitation, and virus inoculation, the input is a weighted and directed social network, where the weight  $\beta_{xy}$  (different from  $\beta_{yx}$ ) corresponds to a measure of influence of node  $x$  on node  $y$ . The input to the preference aggregation problem is a weighted but undirected social network, where the weight  $\beta_{xy}$  (or equivalently  $\beta_{yx}$ ) corresponds to a measure of similarity between preferences of nodes  $x$  and  $y$ . The input to the clustering problem is a dataset of points, which can be converted to a weighted undirected complete network, where the weight  $\beta_{xy}$  (or equivalently  $\beta_{yx}$ ) corresponds to a measure of similarity between data points  $x$  and  $y$ .

## 1.1 Background and Literature

Most of the practical *top-k* problems are computationally hard to solve, even under very simplified models. Certain properties of the underlying objective functions, such as monotonicity and submodularity, have been exploited to develop constant-factor approximation algorithms, for example, greedy hill-climbing for the influence maximization problem under simplified models of information diffusion [7]. However, these properties are lost once the models become more realistic. So there is a need of developing algorithms that are agnostic in practice to the properties of the underlying objective function.

There exist methods for combinatorial optimization, like cross-entropy [3], which work well in practice. However, in recent times, cooperative game theoretic solution concepts have been used effectively in the literature for detecting top  $k$  critical nodes. A cooperative game is represented by  $(N, \nu)$ , where  $N$  is the set of nodes and  $\nu : 2^N \rightarrow \mathbb{R}$  is the valuation function. A solution concept is an allocation  $(\phi_1, \dots, \phi_n)$ , where  $n = |N|$  and  $\phi_x$  is the value allotted to node  $x$ . There exist several solution concepts in the literature, some of the popular ones being Shapley value, Nucleolus, Banzhaf index, Gately point, etc., each possessing an individual list of certain desirable properties. It can be seen that if the valuation function of the game is defined as the objective function itself, the value allotted to a node by most of the popular solution concepts acts as a good measure of appropriateness of that node to be included in the *top-k* set. This has led to application of cooperative game theory for several problems of practical interest.

Narayanam and Narahari [8] propose a heuristic based on Shapley value for influence maximization in social networks, whereas Premraj and Narahari [9] use Shapley value for the problem of influence limitation in social networks. Garg, Narahari, and Murty [6] formulate the clustering problem as a cooperative game and also show that Shapley value for the game can be exactly computed in polynomial time, while Dhamal et al. [4] prove

the equivalence of several solution concepts, namely, Shapley value, Nucleolus, Gately point, and  $\tau$ -value for the formulated game.

## 1.2 Motivation

As mentioned earlier, it has been observed in the literature that the value allotted to a node by most of the well-known cooperative game theoretic solution concepts, acts as a good measure of appropriateness of that node to be included in the *top-k* set. However, it can be easily seen using experiments that the value allotted to a node is as a good measure, when the node has to be chosen all by itself. As the appropriateness of a node to be a part of the *top-k* set depends on other nodes in the set, nodes having the highest  $k$  values are generally not the desirable *top-k* nodes. This fact is not explicitly captured by cooperative game theoretic solution concepts, and so in order to output the suitable *top-k* nodes, it is necessary to post-process the obtained values. Hence the motivation behind this paper is to propose several such post-processing methods. We also propose a standalone algorithm that combines cooperative game theoretic solution concepts with the popular greedy hill-climbing algorithm, taking into account the fact that the appropriateness of a node to be chosen in the *top-k* set depends on other nodes in the set.

We now describe the proposed post-processing methods, primarily in the context of information diffusion; they can be easily extended to other contexts. However, some methods would be more suitable for a given application than others.

## 2 Post-processing Methods

The most direct and naïve method of obtaining the *top-k* set is to sort nodes in descending order of their values, say *ordered\_list*, and then choose the first  $k$  nodes from the list. However, as explained earlier, nodes having the highest  $k$  values are generally not the desirable *top-k* nodes, and so there is a need to post-process the obtained values in order to build an effective *top-k* set. We propose several post-processing methods that can be broadly classified into two types, namely,

- Eliminating neighbors of chosen nodes
- Discounting values of neighbors of chosen nodes

In the following methods, neighbors of a node account for both its in-neighbors and out-neighbors; however, the accounting of neighbors can be altered based on the application. In the methods that follow, in order to obtain an ordering over multiple nodes that are allotted equal values, the ties are broken randomly.

### 2.1 Eliminating neighbors of chosen nodes

As choosing nodes naïvely may result in the *top-k* nodes to be clustered in one part of the network, these methods try to choose the nodes such that they are appropriately spread in the network, so that they influence or represent as many distinct nodes as possible. In all the methods of this type, if it is not possible to choose any more nodes using

the elimination approach, the methods reiterate over *ordered\_list* and naively choose the unchosen nodes in order. It is to be noted that in the methods of this type, in the context of information diffusion where the network is directed, the term *neighbors* refers to both in-neighbors and out-neighbors. Also, if a node  $x$  is both in-neighbor and out-neighbor of a node  $y$ , we consider the mutual edge weight to be  $\max\{\beta_{xy}, \beta_{yx}\}$  whenever applicable.

### 2.1.1 Eliminating neighbors of chosen nodes always

This method is the one used in [8] for influence maximization in a social network. It keeps on choosing the nodes in order from *ordered\_list* and skips a node if any of its neighbors is already chosen in the *top-k* set. This method is observed to perform well in the context of information diffusion since once a node is chosen, it would more or less influence its out-neighbors either directly or indirectly (in multiple hops through other nodes); furthermore, if a node is chosen before its in-neighbors because of its high value, it is likely that its in-neighbors would have a high value only because they influence the former (a more influential node); so the method eliminates both in-neighbors and out-neighbors. Similarly, for the preference aggregation problem, once a node is chosen, it would more or less represent its neighbors well. Note that this method is unsuitable for the clustering problem as the converted input is a complete network.

### 2.1.2 Eliminating neighbors of chosen nodes based on a threshold

This method is similar to the one used in [6] in the context of clustering. The method that eliminates neighbors of chosen nodes always, suffers from the fact that multiple nodes that are highly influential (more or less independent of each other) may be connected with low edge weights; in such cases, it is undesirable to eliminate the neighbors of such influential nodes. So instead of eliminating neighbors of chosen nodes always, this method keeps on choosing the nodes in order from *ordered\_list* and skips a node if any of its neighbors which is already chosen in the *top-k* set, is such that the corresponding edge weight exceeds a certain threshold. This method would work well in all the contexts, provided the threshold is chosen appropriately. One can come up with several variants of this method; for instance, the threshold could be a fixed one for the entire network or dataset [6], or it could be a function of the value of the chosen node itself [4].

### 2.1.3 Eliminating neighbors of chosen nodes based on their local networks

This method determines whether a node should be selected based on its local neighborhood. It keeps on choosing the nodes in order from *ordered\_list* and skips a node  $x$  if there exists its neighbor  $y$  which is already chosen in the *top-k* set such that, when all the neighbors of  $y$  are ordered in decreasing order of their edge weights with  $y$ , then  $x$  lies in the *first half*. Note that this fraction *half* is just a natural first guess; as it acts like a threshold, it can also be a fixed one for the entire network or it can be a function of the value of the chosen node. Intuitively, this method does not eliminate a node when it is a good candidate in the local neighborhood of the already chosen nodes, that is, it is less likely to be influenced by or to influence (or is less similar to) the nodes in the *top-k* set.

## 2.2 Discounting values of neighbors of chosen nodes

The elimination methods are strict owing to their 0/1 nature of eliminating a node. Moreover, it is highly likely that it would not be possible to choose any more nodes using these methods beyond a certain  $k$ , resulting in a naïve selection of the unchosen nodes. One way to overcome these problems is to discount the values of the neighbors of the chosen nodes based on certain criteria instead of eliminating them.

These methods run in  $k$  steps where the value of each node gets updated in each step  $t$ . Let  $top-k^{(t)}$  be the  $top-k$  set in step  $t$ , and let  $\phi_x^{(t)}$  be the value of node  $x$  in step  $t$ . The initializing  $top-k$  set can be given by  $top-k^{(0)} = \{\}$ , and the initializing value of a node  $\phi_x^{(0)} = \phi_x$  is the original value allotted to it. In the methods of this category, we do not update the values of the nodes which are already chosen in the  $top-k$  set, that is,

$$\phi_z^{(t)} = \phi_z^{(t-1)} \quad \forall z \in top-k^{(t-1)} \quad (1)$$

In order to explain the methods of this category, let  $y$  be the node chosen in step  $t$ , that is,  $top-k^{(t)} \setminus top-k^{(t-1)} = \{y\}$ . Also let  $\mathcal{N}_w$  be the set of neighbors of a node  $w$ .

### 2.2.1 Discounting values of neighbors of chosen nodes - I

$$\phi_x^{(t)} = (1 - \beta_{yx}) \phi_x^{(t-1)} \quad \forall x \in \mathcal{N}_y \setminus top-k^{(t-1)} \quad (2)$$

$$\text{or } \phi_x^{(t)} = \left( \prod_{w \in \mathcal{N}_x \cap top-k^{(t)}} (1 - \beta_{wx}) \right) \phi_x^{(0)} \quad (3)$$

This discounting is natural in the independent cascade model of information diffusion where  $\beta_{yx}$  corresponds to the parameter  $p_{yx}$  (in independent cascade model, when node  $y$  first becomes active at time  $\tau$ , it is given a single chance to activate each of its currently inactive neighbor  $x$  at time  $\tau + 1$  and it succeeds with probability  $p_{yx}$ ). Since node  $x$  would get activated because of node  $y$  with probability  $p_{yx}$ , the value of node  $x$  should be discounted by the factor of  $p_{yx}$  whenever any of its neighbor  $y$  gets chosen in the  $top-k$  set. Equivalently, since node  $x$  would not get directly activated by any of its neighbors that are chosen in the  $top-k$  set, with probability  $\prod_{w \in \mathcal{N}_x \cap top-k^{(t)}} (1 - p_{wx})$ , the value of node  $x$  is updated using this factor. Note that we ignore the possibility that the influence of node  $y$  can reach node  $x$  in multiple hops.

### 2.2.2 Discounting values of neighbors of chosen nodes - II

$$\phi_x^{(t)} = \phi_x^{(0)} - \beta_{yx} \phi_x^{(0)} \quad \forall x \in \mathcal{N}_y \setminus top-k^{(t-1)} \quad (4)$$

$$\text{or } \phi_x^{(t)} = \left( 1 - \sum_{w \in \mathcal{N}_x \cap top-k^{(t)}} \beta_{wx} \right) \phi_x^{(0)} \quad (5)$$

This discounting is natural in the linear threshold model of information diffusion where  $\beta_{yx}$  corresponds to the parameter  $b_{yx}$  (in linear threshold model,  $b_{yx}$  is the influence weight of node  $y$  on node  $x$  such that the sum of the influence weights from all of its incoming

neighbors is at most 1; node  $x$  gets activated if the sum of the influence weights from its active incoming neighbors exceeds a certain threshold  $\theta_x$  that is drawn from a uniform distribution between  $[0, 1]$ ). Analogous to the argument in the previous method, since node  $x$  would not get directly activated by any of its neighbors that are chosen in the  $top-k$  set, with probability  $1 - \sum_{w \in \mathcal{N}_x \cap top-k^{(t)}} b_{wx}$ , the value of node  $x$  is updated using this factor. Note that in this method also, we ignore the possibility that the influence of node  $y$  can reach node  $x$  in multiple hops.

### 2.2.3 Discounting values of neighbors of chosen nodes - III

$$\phi_x^{(t)} = \phi_x^{(t-1)} - \beta_{xy} \phi_y^{(t-1)} \quad \forall x \in \mathcal{N}_y \setminus top-k^{(t-1)} \quad (6)$$

$$\text{or } \phi_x^{(t)} = \phi_x^{(0)} - \sum_{w \in \mathcal{N}_x \cap top-k^{(t)}} (\beta_{xw} \phi_w^{(t-1)}) \quad (7)$$

This discounting is natural in both independent cascade and linear threshold models of information diffusion (note the swapping of  $x$  and  $y$  with respect to the previous methods). In the independent cascade model, as node  $x$  influences node  $y$  directly with probability  $p_{xy}$ , it gets a fractional share of the value of node  $y$  (since  $x$  would be influencing other nodes indirectly, through  $y$ ). Now given that node  $y$  is chosen in the  $top-k$  set, the share of  $y$ 's value should be removed from the value of  $x$ . This method uses a simplified expression for this share, namely,  $p_{xy} \phi_y^{(t-1)}$ . Similar argument leads this share to be  $b_{xy} \phi_y^{(t-1)}$  in the linear threshold model. Note that we may be possibly removing more share than required since there may exist multiple neighbors of  $x$ , that are already chosen in the  $top-k$  set, with shares of similar nature (for example, they may be likely to influence almost the same set of nodes). Owing to this, it is possible for the value of a node to become negative.

Furthermore, depending on the application, one may also update the values of the nodes using a suitable convex combination of the aforementioned discounting methods.

## 3 Combining with greedy hill-climbing algorithm

The greedy hill-climbing algorithm is one of the most basic methods used for combinatorial optimization. Starting with the null set, the greedy hill-climbing algorithm selects  $k$  nodes one at a time, each time choosing a node that provides the largest marginal increase in the value of the objective function.

The following method directly utilizes the fact that the appropriateness of a node to be chosen in the  $top-k$  set depends on other nodes which are already chosen in the set. It is to be noted that this method is a standalone algorithm and not a post-processing method.

This method runs in  $k$  steps where in step  $t$ , we define a new cooperative game  $(N \setminus top-k^{(t-1)}, \omega^{(t)})$  where the valuation function  $\omega^{(t)} : 2^{N \setminus top-k^{(t-1)}} \rightarrow \mathbb{R}$  is given by

$$\omega^{(t)}(S) = \nu(top-k^{(t-1)} \cup S) - \nu(top-k^{(t-1)}) \quad (8)$$

Now form  $top-k^{(t)} = top-k^{(t-1)} \cup y$ , where  $y$  is a node with the maximum allocation value in the game  $(N \setminus top-k^{(t-1)}, \omega^{(t)})$ .

### 3.1 Combining Shapley value with greedy hill-climbing

The Shapley value of a node can be computed as the average marginal contribution of the node in all possible permutations over the set of nodes. Though computing exact Shapley value allocation for a general game is hard, it can be approximately computed by considering only a subset of the set of all permutations.

We present a simple algorithm for using Shapley value with greedy hill-climbing for *top-k* problems. The first node to be chosen in the *top-k* set (say  $y_1$ ) can be obtained as the node with the highest Shapley value, where the Shapley value allocation is computed by considering only a subset of the set of all permutations. The second node to be chosen in the *top-k* set (say  $y_2$ ) can be obtained as the node with the highest Shapley value among the set of nodes excluding  $y_1$ , where the Shapley value allocation is now computed by considering only a subset of the set of all permutations in which  $y_1$  is positioned in the first place. In general, the  $\kappa^{th}$  node to be chosen in the *top-k* set (say  $y_\kappa$ ) can be obtained as the node with the highest Shapley value among the set of nodes excluding  $y_1, \dots, y_{\kappa-1}$ , where the Shapley value allocation is computed by considering only a subset of the set of all permutations in which  $y_1, \dots, y_{\kappa-1}$  are positioned in the first  $\kappa - 1$  places.

Note, however, that the running time of the combined algorithm would be approximately  $k$  times the running time of computing the approximate Shapley value allocation. Some of the possible ways of reducing the running time would be to:

- (a) reduce the number of permutations for the computation of approximate Shapley value allocation, since we are only interested in obtaining the node with the highest or near-highest value, and not in the actual allocation itself,
- (b) reduce the number of permutations for later steps, that is, consider more permutations for obtaining  $y_1$  than that for obtaining  $y_2$ , and so on, owing to the inherent fact that obtaining an effective  $y_1$  is more critical than obtaining an effective  $y_2$ ,
- (c) use the combined algorithm only for choosing the first few  $\tilde{k}$  critical nodes and then use some post-processing or discounting approach for choosing the remaining  $k - \tilde{k}$  nodes,
- (d) store the marginal contributions of nodes in permutations which are likely to reoccur, for instance, permutations in which nodes with highest weighted degrees are positioned in the first few places.

## 4 Discussion

We proposed several methods of post-processing the values obtained using any given solution concept from cooperative game theory. The proposed methods are quite generic in nature, but it is possible that some methods better suit a given context than others. One can derive variants of the proposed methods or use multiple methods in conjunction. We also proposed a standalone algorithm that combines cooperative game theoretic solution concepts with the greedy hill-climbing algorithm, by considering that the appropriateness of a node to be chosen in the *top-k* set depends on other nodes in the set.



We compared the proposed methods on small datasets for the purpose of our observations, the results of which are not presented here. The results of extensive experimentation on large datasets will be directly added in the subsequent versions of the paper.

For any practical problem, the natural valuation function would lead to intractable computation for most solution concepts. Though approximate algorithms exist for several solution concepts, an alternative would be to formulate a valuation function that closely resembles the problem and at the same time, facilitates efficient computation of solution concepts.

## References

- [1] Zeinab Abbassi and Hoda Heidari. Toward optimal vaccination strategies for probabilistic models. In *Proceedings of the 20th international conference companion on World wide web*, pages 1–2. ACM, 2011.
- [2] Ceren Budak, Divyakant Agrawal, and Amr El Abbadi. Limiting the spread of misinformation in social networks. In *Proceedings of the 20th international conference on World wide web*, pages 665–674. ACM, 2011.
- [3] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.
- [4] Swapnil Dhamal, Satyanath Bhat, KR Anoop, and Varun R Embar. Pattern clustering using cooperative game theory. *arXiv preprint arXiv:1201.0461*, 2012.
- [5] Swapnil Dhamal and Y Narahari. Scalable preference aggregation in social networks. In *First AAAI Conference on Human Computation and Crowdsourcing*, pages 42–50. AAAI, 2013.
- [6] Vikas K Garg, Y Narahari, and M Narasimha Murty. Novel biobjective clustering (bigc) based on cooperative game theory. *Knowledge and Data Engineering, IEEE Transactions on*, 25(5):1070–1082, 2013.
- [7] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.
- [8] Ramasuri Narayanam and Y Narahari. A shapley value-based approach to discover influential nodes in social networks. *IEEE Transactions on Automation Science and Engineering*, (99):1–18, 2010.
- [9] H Premraj and Y Narahari. Influence limitation in multi-campaign social networks: A shapley value based approach. In *Automation Science and Engineering (CASE), 2012 IEEE International Conference on*, pages 735–740. IEEE, 2012.